Knot removal for tensor product splines

T. Brenna

Dept. of Informatics, Univ. of Oslo, Oslo.
trondbre@ifi.uio.no

Abstract

Given a spline function as a B-spline expansion the object of knot removal is to remove as many knots as possible without perturbing the spline by more than a specified tolerance. In 1987 Lyche and Mørken proposed an efficient knot removal algorithm which determines both the number of remaining knots and their position automatically. In this paper we show how their method can be extended to knot removal techniques for multivariate tensor product splines. We propose a number of new strategies for removing as many knots as possible, and discuss some of the advantages and challenges posed by the special structure of tensor product splines.

1 Introduction

Given a spline function we are often interested in an approximate representation requiring less data. The object of knot removal is to remove as many knots as possible from a given spline without perturbing the spline by more than a given tolerance. An efficient knot removal strategy presented in [6] determines both the number of remaining knots and their location automatically. This strategy was later extended to parametric curves and surfaces in [5], and incorporated with various constraints such as monotonicity and convexity in [1]. An efficient implementation of knot removal for the special case of trilinear splines is given in [3]. In this paper we address some of the questions and problems arising when extending the knot removal technique to multivariate tensor product splines.

The outline of this paper is as follows. We start by fixing notation and presenting techniques for representing tensor product splines. We then proceed with generalizations of coefficient norms, approximation methods, methods for ranking the knots etc., as we review the central parts of the knot removal strategy. Two different ways of performing knot removal are given together with accompanying strategies for finding the desired approximations. We end the paper with two examples demonstrating various aspects of the knot removal techniques presented.

2 Notation

Let $\mathbf{d} = (d_k), \mathbf{m} = (m_k) \in \mathbb{Z}^s$ with $\mathbf{0} \leq \mathbf{d} < \mathbf{m}$ (component-wise) for some positive integer s. Also let $\mathbf{t}^k = \{t_i^k\}_{i=1}^{m_k+d_k+1}$ be a knot vector with $d_k + 1$ equal knots at both ends and with no knot value occurring more than $d_k + 1$ times, for $k = 1, \ldots, s$. In this paper we will treat the collection $\mathbf{t} = \{\mathbf{t}^k\}_{k=1}^{s}$ as a "single" knot vector with "length"

$\mathbf{m}+\mathbf{d}+\mathbf{1}$ defined to be the sum of the length of the knot vectors $\mathbf{t}^k$, $k=1,\ldots,s$. Given such a knot vector we may form products of the basis functions associated with each individual knot vector $\mathbf{t}^k$. By letting

$$B_{\mathbf{i}}(\mathbf{x}) = B_{\mathbf{i},\mathbf{d},\mathbf{t}}(\mathbf{x}) = \prod_{k=1}^{s} B_{i_k,d_k,\mathbf{t}^k}(x_k) \quad \text{for} \quad \mathbf{1} \le \mathbf{i} \le \mathbf{m},$$

where $\mathbf{i} = (i_k) \in \mathbb{Z}^s$ and $\mathbf{x} = (x_k) \in \mathbb{R}^s$,we get a total of $\prod_{k=1}^{s} m_k$ new basis functions for the tensor product space $\mathbb{S}_{\mathbf{d},\mathbf{t}} = \bigotimes_{k=1}^{s} \mathbb{S}_{d_k,\mathbf{t}^k}$. In this paper we let $B_{i_k,d_k,\mathbf{t}^k}$ be the i_kth B-spline of degree d_k associated with $\mathbf{t}^k$, for $k=1,\ldots,s$.

To represent an element of $\mathbb{S}_{\mathbf{d},\mathbf{t}}$ we use a variant of the classical Kronecker product of matrices. Recall that if $\mathbf{A} = (\mathbf{a}_{\mathbf{i},\mathbf{j}})_{\mathbf{i}=\mathbf{1},\mathbf{j}=\mathbf{1}}^{\mathbf{m_1},\mathbf{n_1}} \in \mathbb{R}^{\mathbf{m_1},\mathbf{n_1}}$, $\mathbf{B} = (\mathbf{b}_{\mathbf{i},\mathbf{j}})_{\mathbf{i}=\mathbf{1},\mathbf{j}=\mathbf{1}}^{\mathbf{m_2},\mathbf{n_2}} \in \mathbb{R}^{\mathbf{m_2},\mathbf{n_2}}$ then this product is given by $\mathbf{A}\otimes\mathbf{B} = (\mathbf{a}_{\mathbf{i},\mathbf{j}}\mathbf{B})_{\mathbf{i}=\mathbf{1},\mathbf{j}=\mathbf{1}}^{\mathbf{m_1},\mathbf{n_1}}$. In this paper we will use the "equivalent" product defined by $\mathbf{A}\otimes\mathbf{B} = (\mathbf{A}\mathbf{b}_{\mathbf{i},\mathbf{j}})_{\mathbf{i}=\mathbf{1},\mathbf{j}=\mathbf{1}}^{\mathbf{m_2},\mathbf{n_2}}$, which gives a more convenient ordering of the matrix elements for our use. Also recall that for real matrices $\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}$ we have the following useful relations (assuming that the matrix products and inverses are defined) $(\mathbf{A}\otimes\mathbf{B})(\mathbf{C}\otimes\mathbf{D}) = (\mathbf{AC})\otimes(\mathbf{BD})$, $(\mathbf{A}\otimes\mathbf{B})^{-1} = \mathbf{A}^{-1}\otimes\mathbf{B}^{-1}$ and $\mathbf{A}\otimes\mathbf{B} = \mathbf{P_1}(\mathbf{B}\otimes\mathbf{A})\mathbf{P_2}$, for some permutation matrices $\mathbf{P_1}$ and $\mathbf{P_2}$. In addition we have that the product $\mathbf{A}\otimes\mathbf{B}$ will have linearly independent columns, provided the same holds for $\mathbf{A}$ and $\mathbf{B}$. For further properties of the Kronecker product we refer to [4].

An element

$$f(\mathbf{x}) = \sum_{i_1=1}^{m_1} \cdots \sum_{i_s=1}^{m_s} f_{i_1,\ldots,i_s} \prod_{k=1}^{s} B_{i_k,d_k,\mathbf{t}^k}(x_k) = \sum_{\mathbf{i}\le\mathbf{m}} f_{\mathbf{i}} B_{\mathbf{i},\mathbf{d},\mathbf{t}}(\mathbf{x}) \in \mathbb{S}_{\mathbf{d},\mathbf{t}}$$

can now be written

$$f(\mathbf{x}) = \mathbf{B}_{\mathbf{t}}^{\mathbf{T}}\mathbf{f},$$

where $\mathbf{B_t} = \bigotimes_{\mathbf{k}=\mathbf{1}}^{\mathbf{s}} \mathbf{B}_{\mathbf{t}^k}$ with $\mathbf{B}_{\mathbf{t}^k} = (\mathbf{B}_{\mathbf{1},\mathbf{d_k},\mathbf{t}^k},\ldots,\mathbf{B}_{\mathbf{m_k},\mathbf{d_k},\mathbf{t}^k})^{\mathbf{T}}$ for $k=1,\ldots,s$. Here $\mathbf{f}$ is a vector containing the B-spline coefficients $\mathbf{F} = (\mathbf{f}_{\mathbf{i_1},\ldots,\mathbf{i_s}})$ of f given by $\mathbf{f} = \mathbf{vec}(\mathbf{F}) := \sum_{\mathbf{i}\le\mathbf{m}} \mathbf{f_i}\mathbf{e_i}$, where $\mathbf{e_i} = \bigotimes_{k=1}^{s} \mathbf{e}_{i_k}$ with $\mathbf{e}_{i_k} \in \mathbb{R}^{m_k}$. Finally we state that for a tensor of real coefficients $\mathbf{F} = (\mathbf{f_i})_{\mathbf{1}\le\mathbf{i}\le\mathbf{m}} \in \mathbb{R}^{\mathbf{m}}$ we let $\mathbf{F}^{(\sigma_k)}$ denote the tensor $\mathbf{F}$ with its elements rearranged according to the cyclic permutation of the s-tuple $\{1,2,\ldots,s\}$ given by $\boldsymbol{\sigma}_k = \{k,k+1,\ldots,s,1,\ldots,k-1\}$, for $k=1,\ldots,s$.

Finally, for a spline $f = \sum_{\mathbf{i}\le\mathbf{m}} f_{\mathbf{i}} B_{\mathbf{i},\mathbf{d},\mathbf{t}}(\mathbf{x})$ we define a class of weighted l^p-norms of its B-spline coefficients, given by

$$\| f \|_{l^p,\mathbf{t}} = \begin{cases} (\sum_{\mathbf{i}\le\mathbf{m}} w_{\mathbf{i}} |f_{\mathbf{i}}|^p)^{1/p}, & \text{for} \quad 1 \le p < \infty, \\ \max\limits_{\mathbf{1}\le\mathbf{i}\le\mathbf{m}} |f_{\mathbf{i}}|, & \text{for} \quad p = \infty, \end{cases}$$

where the weights are given by $w_{\mathbf{i}} = \prod_{k=1}^{s} \frac{t^k_{i_k+d_k+1} - t^k_{i_k}}{d_k+1}$, for $\mathbf{1} \le \mathbf{i} \le \mathbf{m}$. Using the notation introduced above we have that $\| f \|_{l^p,\mathbf{t}} = \| \mathbf{W}_{\mathbf{t}}^{1/\mathbf{p}}\mathbf{f} \|_{\mathbf{l^p}}$, $(p \ge 1)$ where $\mathbf{W_t}$ is a

diagonal scaling matrix given by

$$\mathbf{W_t} = \bigotimes_{\mathbf{k=1}}^{\mathbf{s}} \mathbf{W_{t^k}}, \quad \text{with} \quad \mathbf{W_{t^k}} = \mathbf{diag}\left(\left(\frac{\mathbf{t^k_{d_k+2}} - \mathbf{t^k_1}}{\mathbf{d_k}+1}\right), \ldots, \left(\frac{\mathbf{t^k_{m_k+d_k+1}} - \mathbf{t^k_{m_k}}}{\mathbf{d_k}+1}\right)\right).$$

These coefficient norms are easy to compute and are known to approximate the ordinary L^p-norms well for splines of moderate degree [2,6]. In the algorithms we use $p = 2$ when computing approximations and $p = \infty$ to measure the error.

3 The knot removal algorithm

Given an element $f \in \mathbb{S}_{\mathbf{d},\mathbf{t}}$, a tolerance $\varepsilon > 0$ and some norm $\|\cdot\|$ the goal of the knot removal algorithm presented in [6] is to find a subspace $\mathbb{S}_{\mathbf{d},\boldsymbol{\tau}}$ of $\mathbb{S}_{\mathbf{d},\mathbf{t}}$ ($\boldsymbol{\tau} \subseteq \mathbf{t}$) and an element $g \in \mathbb{S}_{\mathbf{d},\boldsymbol{\tau}}$ with $\| f - g \| < \varepsilon$, and where we want $\boldsymbol{\tau}$ to be of minimal length. In this section we review the basic parts of this algorithm as we extend the theory to tensor product splines. Further details of the material in this section can be found in [2].

3.1 Finding approximations

To approximate $f \in \mathbb{S}_{\mathbf{d},\mathbf{t}}$ in a subspace $\mathbb{S}_{\mathbf{d},\boldsymbol{\tau}}$, where $\boldsymbol{\tau}$ is of "length" $\mathbf{n} + \mathbf{d} + \mathbf{1}$ with $\mathbf{n} \leq \mathbf{m}$, we use the spline g which is the best approximation to f in the $l^2, \mathbf{t}$-norm. In other words, the spline we seek will be the solution to the minimization problem $\min\limits_{h \in \mathbb{S}_{\mathbf{d},\boldsymbol{\tau}}} \| f - h \|^2_{l^2,\mathbf{t}}$. Solving this problem is equivalent to solving the linear least squares problem given by

$$\min_{\mathbf{C} \in \mathbb{R}^{\mathbf{n}}} \| \mathbf{W_t^{1/2}} (\mathbf{Ac} - \mathbf{f}) \|^2_{\mathbf{l^2}}, \tag{3.1}$$

where $\mathbf{A} = \bigotimes_{\mathbf{k=1}}^{\mathbf{s}} \mathbf{A_k}$ is the knot insertion matrix from $\boldsymbol{\tau}$ to $\mathbf{t}$ (i.e. $\mathbf{A_k}$ is the knot insertion matrix from $\boldsymbol{\tau}^k$ to $\mathbf{t}^k$, for $k = 1, \ldots, s$), $\mathbf{f} = \mathbf{vec}(\mathbf{F})$ are the given B-spline coefficients of f in $\mathbb{S}_{\mathbf{d},\mathbf{t}}$ and $\mathbf{c} = \mathbf{vec}(\mathbf{C})$ are the unknown B-spline coefficients of g in $\mathbb{S}_{\mathbf{d},\boldsymbol{\tau}}$. Since the knot insertion matrix $\mathbf{A}$ has full rank and $\mathbf{W_t}$ is non-singular, the normal equations $\mathbf{A^T W_t A c} = \mathbf{A^T W_t f}$ associated with the system (3.1) will have a unique solution which can be found ([2,3]) by solving a series of s tensor equation systems given by

$$(\mathbf{A_k^T W_{t^k} A_k}) \mathbf{D_k^{(\sigma_k)}} = (\mathbf{A_k^T W_{t^k}}) \mathbf{D_{k-1}^{(\sigma_k)}}, \tag{3.2}$$

for $k = 1, \ldots, s$. Here $\mathbf{D_k} \in \mathbb{R}^{\mathbf{n_k}}$ with $\mathbf{n}_k = (n_1, \ldots, n_k, m_{k+1}, \ldots, m_s)$, and we let $\mathbf{D_0} = \mathbf{F}$, and set the coefficients of the approximation g equal to the solution of the last tensor equation system, $\mathbf{C} = \mathbf{D_s}$. The tensor equations (3.2) can be efficiently solved by calculating the Cholesky factorization of the banded coefficient matrix $(\mathbf{A_k^T W_{t^k} A_k})$ and solving for each right hand side in the tensor $(\mathbf{A_k^T W_{t^k}}) \mathbf{D_{k-1}^{(\sigma_k)}}$.

3.2 Ranking the knots

The final approximation to the initial spline is found by searching through a sequence of approximations, constructed by using the approximation method of the previous section, on subsets of the knots of the initial spline. These subsets are calculated by associating a weight with each interior knot, representing a rough measure of its importance. See [6] for

the details. For higher dimensional tensor product splines we set the weight for a given knot to the maximum of the weights corresponding to this knot when the calculation is iterated over the "remaining" parameter directions. We refer to [2] for further details.

4 Knot removal methods

When removing knots from a tensor product spline we are faced with more options than in the case of a spline curve. In this section we present two different ways of performing knot removal. The first one studied in [2] based on a symmetric approach, treats all the parameter directions of a tensor product spline simultaneously, while the second one will treat one parameter direction at a time.

4.1 Knot removal based on a symmetric approach

If we let $G_f(\boldsymbol{\tau})$ denote the approximation to $f \in \mathbb{S}_{\mathbf{d},\mathbf{t}}$ defined on the knot vector $\boldsymbol{\tau}$ we see that the approximations in the sequence mentioned above can be written $\{G_f(\boldsymbol{\tau}_j)\}_{j=0}^{N}$, where $\boldsymbol{\tau}_j$ is constructed from $\mathbf{t}$ by removing j of its interior knots, and $N = \sum_{k=1}^{s}[m_k - (d_k + 1)]$ is the total number of interior knots of $\mathbf{t}$. Given such a sequence of approximations we can perform a search on the index j to determine an approximation $g^* = G_f(\boldsymbol{\tau}^*)$ to the initial spline f with a preferably short knot vector $\boldsymbol{\tau}^*$, and with the property that $\| f - g^* \|_{l^\infty,\mathbf{t}} \leq \varepsilon$, where ε is the specified tolerance. If the knot vector $\boldsymbol{\tau}^*$ is not equal to any of the two knot vectors $\boldsymbol{\tau}_0$ or $\boldsymbol{\tau}_N$ we may repeat the process to find a new approximation based on g^* as proposed in [6]. Taking into account how the sequence $\{G_f(\boldsymbol{\tau}_j)\}_{j=0}^{N}$ was constructed we expect the error $\| f - G_f(\boldsymbol{\tau}_j) \|_{l^\infty,\mathbf{t}}$ to decrease, but not necessarily strictly, for decreasing values of the search parameter j. How the search among the possible approximations is done will generally depend on a number of factors, including some which will be discussed later through examples. Also note that we only have to compute approximations for indexes actually used in the search. By treating all the directions simultaneously we take into consideration the inherent symmetry of the problem. As we will see later this will in some cases enable us to remove more knots than by treating one parameter direction at a time, but it will also lead to more complicated and slower code in an implementation.

4.2 Knot removal for one parameter direction at a time

In the second knot removal method we start by thinking of a spline $f \in \mathbb{S}_{\mathbf{d},\mathbf{t}}$ as a series of parametric curves in corresponding high dimensional spaces. We can then perform a parametric knot removal for each parameter direction. The advantage of this approach is that it is easy to implement since we may use existing knot removal routines for spline curves with only minor modifications.

In the following discussion we let $\varepsilon = \sum_{i=1}^{s} \varepsilon_i$, with $\varepsilon_i \geq 0$ for all i, be a given tolerance. Also let $f(\mathbf{x}) = \sum_{\mathbf{i} \leq \mathbf{m}} f_{\mathbf{i}} B_{\mathbf{i},\mathbf{d},\mathbf{t}}(\mathbf{x}) = \mathbf{B}_{\mathbf{t}}^{\mathbf{T}}\mathbf{f}$ be a spline in $\mathbb{S}_{\mathbf{d},\mathbf{t}} = \bigotimes_{k=1}^{s} \mathbb{S}_{d_k,\mathbf{t}^k}$, with $\mathbf{B}_{\mathbf{t}}^{\mathbf{T}} = \bigotimes_{\mathbf{k}=\mathbf{1}}^{\mathbf{s}} \mathbf{B}_{\mathbf{t}^k}^{\mathbf{T}}$ and $\mathbf{f} = \mathbf{vec}(\mathbf{F})$. We start by identifying a series of parametric curves which may be naturally associated with this tensor product spline. We say that the spline f consists of the curves $\vec{f_k}(x_k)$, for $k = 1, \ldots, s$, where $\vec{f_k}(x_k)$ is the parametric

curve in $\mathbb{R}^{M_k}$, for $M_k = (\prod_{p=1}^{k-1} m_k)(\prod_{p=k+1}^{s} m_k)$, given by

$$\vec{f}_k(x_k) = \Big[\Big(\overset{k-1}{\underset{l=1}{\otimes}} \mathbf{I}_{\mathbf{m_l}}\Big) \otimes \mathbf{B}_{\mathbf{t^k}}^{\mathbf{T}} \otimes \Big(\overset{\mathbf{s}}{\underset{\mathbf{l=k+1}}{\otimes}} \mathbf{I}_{\mathbf{m_l}}\Big)\Big]\mathbf{f}.$$

We now return to the problem of finding a preferably short knot vector $\boldsymbol{\tau} \subseteq \mathbf{t}$ and a spline $g(\mathbf{x}) = \sum_{\mathbf{j}\leq\mathbf{n}} c_{\mathbf{j}} B_{\mathbf{j},\mathbf{d},\boldsymbol{\tau}}(\mathbf{x}) \in \mathbb{S}_{\mathbf{d},\boldsymbol{\tau}} = \overset{s}{\underset{k=1}{\otimes}} \mathbb{S}_{d_k,\boldsymbol{\tau}^k}$ with the property that $\| f - g \|_{l^\infty,\mathbf{t}} \leq \varepsilon$. To apply knot removal to $f \in \mathbb{S}_{\mathbf{d},\mathbf{t}}$ we can now go through the following steps for $k = 1, \ldots, s$.

1. Apply parametric knot removal with the tolerance ε_k to the parametric curve
$$\vec{f}_k(x_k) = \Big[\Big(\overset{k-1}{\underset{l=1}{\otimes}} \mathbf{I}_{\mathbf{n_l}}\Big) \otimes \mathbf{B}_{\mathbf{t^k}}^{\mathbf{T}} \otimes \Big(\overset{\mathbf{s}}{\underset{\mathbf{l=k+1}}{\otimes}} \mathbf{I}_{\mathbf{m_l}}\Big)\Big]\mathbf{f_{k-1}},$$
defined on $\mathbf{t}^k$, starting with $\mathbf{f}_0 = \mathbf{f}$.
2. This will produce a new parametric curve defined on the knot vector $\boldsymbol{\tau}^k \subseteq \mathbf{t}^k$
$$\vec{f}'_k(x_k) = \Big[\Big(\overset{k-1}{\underset{l=1}{\otimes}} \mathbf{I}_{\mathbf{n_l}}\Big) \otimes \mathbf{B}_{\boldsymbol{\tau}^{\mathbf{k}}}^{\mathbf{T}} \otimes \Big(\overset{\mathbf{s}}{\underset{\mathbf{l=k+1}}{\otimes}} \mathbf{I}_{\mathbf{m_l}}\Big)\Big]\mathbf{f_k},$$
where $\mathbf{f}_k = \mathbf{vec}(\mathbf{F_k})$ for $\mathbf{F_k} \in \mathbb{R}^{\mathbf{n_1},\ldots,\mathbf{n_k},\mathbf{m_{k+1}},\ldots,\mathbf{m_s}}$.
3. We also have that
$$\begin{aligned}\vec{f}'_k(x_k) &= \Big[\Big(\overset{k-1}{\underset{l=1}{\otimes}} \mathbf{I}_{\mathbf{n_l}}\Big) \otimes \mathbf{B}_{\boldsymbol{\tau}^{\mathbf{k}}}^{\mathbf{T}} \otimes \Big(\overset{\mathbf{s}}{\underset{\mathbf{l=k+1}}{\otimes}} \mathbf{I}_{\mathbf{m_l}}\Big)\Big]\mathbf{f_k} \\ &= \Big[\Big(\overset{k-1}{\underset{l=1}{\otimes}} \mathbf{I}_{\mathbf{n_l}}\Big) \otimes \mathbf{B}_{\mathbf{t^k}}^{\mathbf{T}} \otimes \Big(\overset{\mathbf{s}}{\underset{\mathbf{l=k+1}}{\otimes}} \mathbf{I}_{\mathbf{m_l}}\Big)\Big]\Big[\Big(\overset{\mathbf{k-1}}{\underset{\mathbf{l=1}}{\otimes}} \mathbf{I}_{\mathbf{n_l}}\Big) \otimes \mathbf{A_k} \otimes \Big(\overset{\mathbf{s}}{\underset{\mathbf{l=k+1}}{\otimes}} \mathbf{I}_{\mathbf{m_l}}\Big)\Big]\mathbf{f_k},\end{aligned}$$
where $\mathbf{A_k}$ is the knot insertion matrix from $\boldsymbol{\tau}^k$ to $\mathbf{t}^k$.
4. And consequently
$$\| \vec{f}_k - \vec{f}'_k \|_{l^\infty,\mathbf{t}^k} = \Big\| \mathbf{f}_{k-1} - \Big[\Big(\overset{k-1}{\underset{l=1}{\otimes}} \mathbf{I}_{\mathbf{n_l}}\Big) \otimes \mathbf{A_k} \otimes \Big(\overset{\mathbf{s}}{\underset{\mathbf{l=k+1}}{\otimes}} \mathbf{I}_{\mathbf{m_l}}\Big)\Big]\mathbf{f_k} \Big\|_{l^\infty} \leq \varepsilon_k.$$

Finally we let the coefficients of the function $g(\mathbf{x}) = \mathbf{B}_{\boldsymbol{\tau}}^{\mathbf{T}}\mathbf{c} \in \mathbb{S}_{\mathbf{d},\boldsymbol{\tau}}$ be $\mathbf{c} = \mathbf{vec}(\mathbf{F_s})$, and we have the following result.

Theorem 4.1 *If we let $f(\mathbf{x}) = \mathbf{B}_{\mathbf{t}}^{\mathbf{T}}\mathbf{f} \in \mathbb{S}_{\mathbf{d},\mathbf{t}}$ and $g(\mathbf{x}) = \mathbf{B}_{\boldsymbol{\tau}}^{\mathbf{T}}\mathbf{c} \in \mathbb{S}_{\mathbf{d},\boldsymbol{\tau}}$ be the tensor product splines from the discussion above, then we have $\| f - g \|_{l^\infty,\mathbf{t}} \leq \varepsilon$.*

Proof: Let $\mathbf{A} = \overset{\mathbf{s}}{\underset{\mathbf{k=1}}{\otimes}} \mathbf{A_k}$ be the knot insertion matrix from $\boldsymbol{\tau}$ to $\mathbf{t}$, and let $f_0(\mathbf{x}) = \mathbf{B}_{\mathbf{t}}^{\mathbf{T}}\mathbf{f_0}$

be equal to f and $f_s(\mathbf{x}) = \mathbf{B}_\tau^{\mathbf{T}}\mathbf{f_s}$ be equal to g, i.e. $\mathbf{f}_0 = \mathbf{f}$ and $\mathbf{f}_s = \mathbf{c}$. Then

$$\begin{aligned}
\| f - g \|_{l^\infty,\mathbf{t}} &= \|\mathbf{f}_0 - \mathbf{Af_s}\|_{l^\infty} \\
&= \left\| \mathbf{f}_0 + \sum_{k=2}^{s} \left(\left[\left(\bigotimes_{l=1}^{k-1} \mathbf{A}_l \right) \otimes \left(\bigotimes_{\mathbf{l=k}}^{\mathbf{s}} \mathbf{I}_{\mathbf{m}_l} \right) \right] \mathbf{f_{k-1}} - \left[\left(\bigotimes_{\mathbf{l=1}}^{\mathbf{k-1}} \mathbf{A}_l \right) \otimes \left(\bigotimes_{\mathbf{l=k}}^{\mathbf{s}} \mathbf{I}_{\mathbf{m}_l} \right) \right] \mathbf{f_{k-1}} \right) - \bigotimes_{\mathbf{k=1}}^{\mathbf{s}} \mathbf{A_k f_s} \right\|_{l^\infty} \\
&\le \sum_{k=1}^{s} \left\| \left[\left(\bigotimes_{l=1}^{k-1} \mathbf{A}_l \right) \otimes \left(\bigotimes_{\mathbf{l=k}}^{\mathbf{s}} \mathbf{I}_{\mathbf{m}_l} \right) \right] \left[\mathbf{f_{k-1}} - \left[\left(\bigotimes_{\mathbf{l=1}}^{\mathbf{k-1}} \mathbf{I}_{\mathbf{n}_l} \right) \otimes \mathbf{A_k} \otimes \left(\bigotimes_{\mathbf{l=k+1}}^{\mathbf{s}} \mathbf{I}_{\mathbf{m}_l} \right) \right] \mathbf{f_k} \right] \right\|_{l^\infty} \\
&\le \sum_{k=1}^{s} \left\| \mathbf{f}_{k-1} - \left[\left(\bigotimes_{l=1}^{k-1} \mathbf{I}_{\mathbf{n}_l} \right) \otimes \mathbf{A_k} \otimes \left(\bigotimes_{\mathbf{l=k+1}}^{\mathbf{s}} \mathbf{I}_{\mathbf{m}_l} \right) \right] \mathbf{f_k} \right\|_{l^\infty} = \sum_{k=1}^{s} \left\| \vec{f_k} - \vec{f_k'} \right\|_{l^\infty,\mathbf{t}^k} \le \varepsilon. \qquad \square
\end{aligned}$$

5 Examples

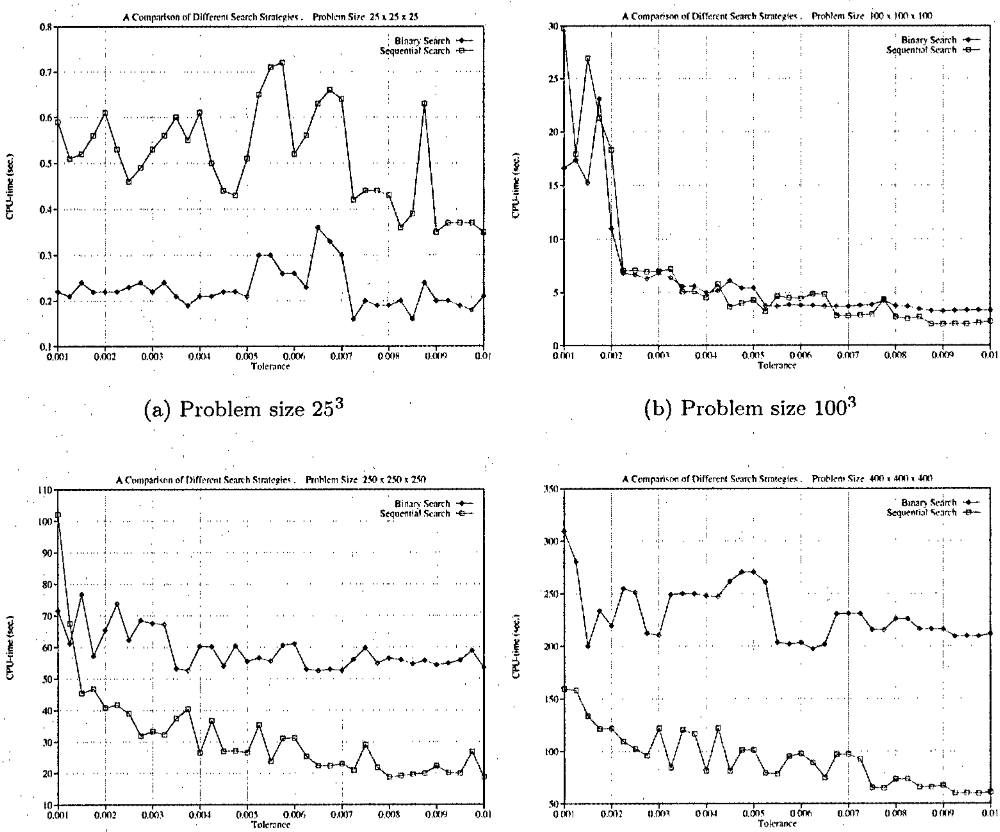
The knot removal methods presented above have been implemented and tested on a computer. In this section we present trivariate examples from this implementation and propose different knot removal strategies depending on the problem at hand. See [3] for a detailed description of this implementation.

Example 5.1 In this first example we will compare two different strategies for searching through a list of approximations $\{G_f(\boldsymbol{\tau}_j)\}_{j=0}^{N}$ introduced above. We will consider the knot removal method treating one parameter direction at a time, which means that we end up solving a parametric knot removal problem with tolerance $\varepsilon_i = \varepsilon/3$, $i = 1, 2, 3$, for each of the three parameter directions.

To improve efficiency the parametric knot removal routine implemented is constructed in a way that lets it abort the computation if an approximation for any component of the parametric curve fails to lie within the specified tolerance. This fact suggests a search strategy where we compute successive approximations to the initial spline by adding one interior knot at a time, starting with zero interior knots, and where each intermediate approximation is given by the first of these approximation processes to be completed. Intuitively we would expect such a *sequential search strategy* to perform best for "large" tolerances and/or large problems, where it is more to gain by aborting an approximation process. In this example we have compared this search strategy with a strategy proposed in [6] using a binary search.

In all the tests we have used an initial trilinear spline constructed by sampling the function given by $f(x, y, z) = \frac{1}{3}[\sin(2\pi x) + \sin(2\pi y) + \sin(2\pi z)]$ in the points specified by a uniform 3-dimensional grid on the domain $\Omega = [0, 1]^3$, for four selected grid sizes. Each spline was reduced by using both of the search strategies mentioned above, for tolerances varying from $\varepsilon = 0.001$ to $\varepsilon = 0.01$. Both of the search strategies produced approximately the same end grid size in each test.

In Figure 1 the CPU-time of the two search strategies is plotted against the tolerance for the selected grid sizes. We observe that the reductions utilizing a binary search perform best on small problems, while the sequential search strategy turn out to be superior for large problems.



(a) Problem size 25^3

(b) Problem size 100^3

(c) Problem size 250^3

(d) Problem size 400^3

FIG. 1. A comparison of two different search strategies.

Example 5.2 In this example we compare the two different knot removal methods presented in this paper. Here we have used an initial trilinear spline constructed by sampling a function given by $f(x, y, z) = e^{\sin(2\pi x^2 yz)}$ in the points specified by a uniform 3-dimensional grid on the domain $\Omega = [0, 1]^3$, for varying grid sizes. Each spline was reduced by both the method based on the symmetric approach and the method treating one parameter direction at a time.

The results are presented in Table 1. We see that in our implementation the method using the symmetric approach is by far the slowest method. However, at least for the type of function considered in this example the method based on the symmetric approach will give a much better reduction than the other.

Knot Removal for Trilinear Splines, Tolerance $\varepsilon = 0.005$						
Start	Parametric, binary search			Symmetric, binary search		
grid	CPU	End grid	Error	CPU	End grid	Error
100^3	16.53	$72 \times 65 \times 65$	$4.93800 \cdot 10^{-3}$	63.23	$54 \times 53 \times 53$	$4.92080 \cdot 10^{-3}$
150^3	56.44	$81 \times 71 \times 71$	$4.80243 \cdot 10^{-3}$	122.2	$51 \times 49 \times 49$	$4.77236 \cdot 10^{-3}$
200^3	99.48	$68 \times 66 \times 66$	$4.91142 \cdot 10^{-3}$	300.9	$54 \times 50 \times 51$	$4.98275 \cdot 10^{-3}$
250^3	165.3	$74 \times 62 \times 62$	$4.74970 \cdot 10^{-3}$	584.8	$61 \times 56 \times 56$	$4.85916 \cdot 10^{-3}$
300^3	256.8	$72 \times 62 \times 62$	$4.85316 \cdot 10^{-3}$	1094	$60 \times 54 \times 53$	$4.81551 \cdot 10^{-3}$
350^3	391.4	$75 \times 65 \times 63$	$4.77028 \cdot 10^{-3}$	1312	$54 \times 50 \times 50$	$4.92422 \cdot 10^{-3}$
400^3	494.6	$71 \times 59 \times 63$	$4.79631 \cdot 10^{-3}$	1865	$54 \times 50 \times 50$	$4.81064 \cdot 10^{-3}$

TAB. 1 Knot removal for the trilinear splines of Example 2.

Bibliography

1. Arge, E., Dæhlen, M., Lyche, T. and Mørken, K. (1990). *Constrained spline approximation of functions and data based on constrained knot removal.* In: *Algorithms for Approximation II*, J. C. Mason and M. G. Cox (eds.), Chapman and Hall, London, 4–20
2. Brenna, T. (1998). *Knot removal for multivariate tensor product splines.* Master thesis, part I. Dept. of Informatics, Univ. of Oslo.
3. Brenna, T. (1998). *Knot removal for linear, bilinear and trilinear splines.* Master thesis, part II. Dept. of Informatics, Univ. of Oslo.
4. Graham, A. (1981). *Kronecker Products and Matrix Calculus With Applications.* Ellis Horwood Series. Mathematics and its applications.
5. Lyche, T. and Mørken, K. (1986). *Knot removal for parametric B-spline curves and surfaces.* Computer Aided Geometric Design, 4, 217–230
6. Lyche, T. and Mørken, K. (1987). *A data reduction strategy for splines with applications to the approximation of functions and data.* IMA Journal of Numerical Analysis, 8, 185–208.